32002

# Use of Data Description Languages in the Interchange of Data

Authors: M.Pignède, B.Real-Planells; European Space Operations Centre (ESOC), Robert Bosch Strasse 5, 64293 Darmstadt, Germany; Tel: +49 6151 902216

S.R.Smith; Logica UK Ltd, 75 Hampstead Road, London NW1 2NT, England; Tel: +44 71 637 9111

## ABSTRACT

The Consultative Committee for Space Data Systems (CCSDS) is developing Standards for the interchange of information between systems, including those operating under different environments. The objective is to perform the interchange automatically, i.e. in a computer interpretable manner. One aspect of the concept developed by CCSDS is the use of a separate data description to specify the data being transferred. Using the description, data can then be automatically parsed by the receiving computer. With a suitably expressive Data Description Language (DDL), data formats of arbitrary complexity can be handled.

The advantages of this approach are that (a) the description need only be written and distributed once to all users (b) new software does not need to be written for each new format, provided generic tools are available to support writing and interpretation of descriptions and the associated data instances. Consequently the effort of "hard coding" each new format is avoided and problems of integrating multiple implementations of a given format by different users are avoided. The approach is applicable in any context where computer parsable description of data could enhance efficiency (e.g. within a spacecraft control system, a data delivery system or an archive).

The CCSDS have identified several candidate DDLs: EAST (Extended Ada Subset), TSDN (Transfer Syntax Data Notation) and **MADEL** (Modified **A**SN.1 as a Data **D**escription **L**anguage -- a DDL based on the Abstract Syntax Notation One - ASN.1 - specified in the ISO/IEC 8824).

This paper concentrates on ESA's development of MADEL. ESA have also developed a "proof of concept" prototype of the required support tools, implemented on a PC under MS-DOS, which has successfully demonstrated the feasibility of the approach, including the capability within an application of retrieving and displaying particular data elements, given its MADEL description (i.e. a data description written in MADEL).

This paper outlines the work done to date and assesses the applicability of this modified ASN.1 as a DDL. The feasibility of the approach is illustrated with several examples.

Keywords: **Heterogeneous Environments, Automated Interchange, Data Description, Modified ASN.1, Demonstrated Feasibility**

## The Problems of Interchanging Data

The problems of data interchange are primarily those associated with providing the destination (potentially a different computer environment) with **all** the information it needs to be able to interpret the received data. At present, a typical data interchange system is dedicated to a particular flight mission or project. Data is acquired, processed, shared to some level with other members of the investigating team and eventually archived. Further, documentation of the data and its format may not be complete and up-to-date. This practice results in the need for a different interchange data system for each mission and makes the reuse of data and software at some future period difficult.

## Overview of the CCSDS Approach

The CCSDS approach is to provide standardised techniques for the automated interpreting of data products in a heterogeneous computer environment. It puts no constraint on the format of the user data and can thus accommodate formats developed by other organisations or user communities. It offers a data labelling scheme

which permits associating a data instance and its (in principle separate) complete and unambiguous description. Further, it allows the development of generic software to support the retrieval, access, parsing and presentation of data to satisfy particular application and user needs. Three major stages are identified in the data interchange process and are explained in this paper.

### The Data Interchange Requirements

To achieve the stated aims, the fundamental requirement that has to be fulfilled is the unambiguous description of data that has to be interchanged between users separated by both time and space. When received, data have to be understood fully. This means that the receiver has to not only be able to read the data from the transfer media and understand the basic physical elements of the data (e.g. an integer), but also the receiver must be able to understand the real world meaning of the data. For example, there is no point in the receiver knowing that the $10^{th}$ and $11^{th}$ bytes of the received data are an *integer of value 145*, if he also does not know that this conveys the *temperature of a spacecraft instrument in degrees Kelvin.*

Further, it is desirable that data products be automatically transferable without requiring any conversion of their contents. It is also desirable that transfers would be possible regardless of the source or destination computer types. In other words,

- the data must remain in its original form, i.e. does **not** need to be converted in order to be interpreted at the destination;
- the destination needs to know nothing about the source, regardless of how different the computer systems may be.

Thus, the approach presented in this paper is to use a data description processable on any computer which will allow the transmission of data in its native form, i.e. no data encoding will be required.

### The Three Stages of the Data Interchange Model

In an attempt to establish a logical model of the whole data description process, an initial assumption was made, that is that the description of data can be cleanly split between the physical description (called the **syntax description**) and the meaning of those physical elements (called the **semantic description**). In fact, as the problem domain was studied further, it was realised that the description should be split into 3 parts, the same physical description as originally perceived, but the meaning component was really 2 separate components: the meaning of the physical elements being exchanged and furthermore the methods of combining the physical elements or relationships between these elements. Indeed, a generator of data products not only wishes to convey the physical data and its meaning, but also how they are intended to be interpreted taking into account their context. Figure 1 shows the model defined by the CCSDS to represent the following stages of data description:

**Stage 1**: Data is initially perceived as a group of bits accessible from a physical medium and is read in combination with its syntax description. This tells the user the physical layout of the data on the medium, which bits are grouped with each other and how they should be read by common computer hardware (e.g. as integers, reals, bit masks, etc). The syntax description must include all the physical bits, all the bits and groupings must be named in some way. At this stage it is possible to manipulate the basic values, for example converting to another physical representation, e.g. from VAX reals to IBM reals.

**Stage 2**: The next stage is when the physical data is read in conjunction with its basic semantic description: this only describes those parts of the data that the user is interested in fully understanding, i.e. the basic semantic

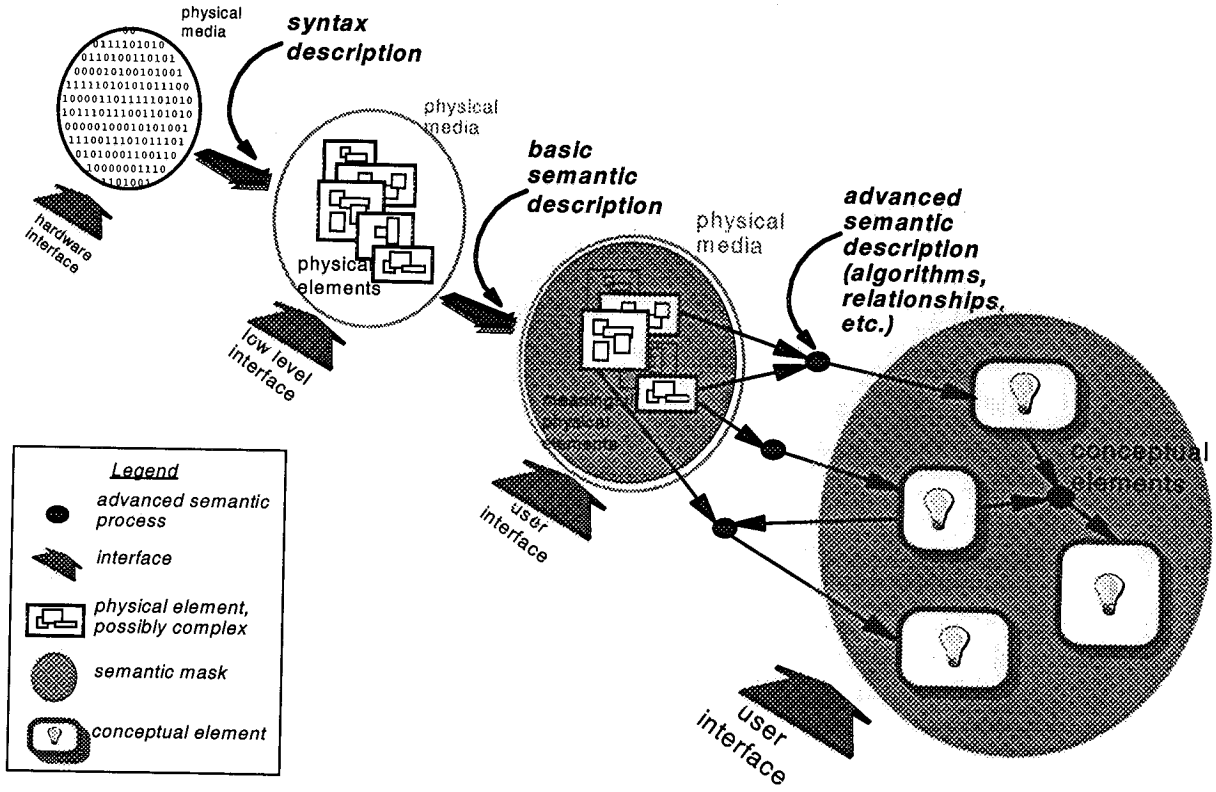


Figure 1: **Data Interchange Logical Model**

description could be perceived as a filter that covers the physical layout of the data and only allows those elements that are of interest to be seen. In this stage, the relevant information is typically: meaning, units, aliases, scaling factor. The elements which are not called out in the basic semantic description are still physically present, but not of interest to the user. Different users may use different semantic descriptions which will act as different masks over the same physical data and therefore perceive the data product in a different manner. Interfacing to the data at this level is the most common method within present day data processing systems.

**Stage 3**: The final stage in the understanding of the data is to add advanced semantics: these do not define the static meaning of each piece of data, but the way different pieces of data are related to each other. This may be specified in a natural language or as a mathematical algorithm or as a process defined in a programmable computer language. The elements thus created are called *conceptual elements*, as they never actually physically exist: they are pieces of information that are carried within an application domain. If they were ever written as a piece of physical data to a physical media, then this whole process would start from the beginning, i.e. with applying their syntax description. Figure 1 shows a ● at the point where the advanced semantics are applied, this is to indicate that some form of processing takes place. For example, the data may contain two integers, named *mass* and *volume* and for these named physical elements the advanced semantic description may also have a definition of *density*, as being a relationship between the two physical elements mass and volume.

## Applicability of MADEL as a DDL

As the model shows, a mechanism must exist at each stage in order to link the data perceived so far to the syntax or semantic information being added. This paper focuses on stage 1: MADEL has been developed to perform the syntactic description of the physical data. Regarding stage 2, it is noted that CCSDS has developed a DDL

called PVL (Parameter Value Language) suited to handle the basic semantic description (see Reference-2). Regarding stage 3, processing is in practice very much application specific. Stages 2 and 3 are not covered further in this paper.

The reasons for selecting ASN.1 (see Reference-1) as a starting point of the MADEL development are that ASN.1 is an International Standard, is familiar to the CCSDS community and intuitive to understand, even by the non expert users. It must be noted at this point that ASN.1 is used as a separate DDL for describing data formats and not as it was originally designed that is, embedding within the data to be transferred auxiliary information ("ASN.1 encoding"). Hence, in the process of deriving MADEL, limitations were made as were some extensions added in order to fulfil the requirements as stated earlier. The most significant modifications to the ISO 8824 ASN.1 specification are detailed below:

**Physical Representation of Base Types**

Typical MADEL description statements are `SpacecraftID ::= OCTET STRING` or `Width ::= REAL`. Since the intention is to keep the physical data in its **native** form and to have the MADEL description in accordance with Reference-1, defaults must be adopted everywhere a complete description of the data is not supplied directly by the MADEL description itself. For example how many octets are in `SpacecraftID` and how many bits are used for the mantissa of `Width` and where are these bits located within the real number layout? These pieces of information must be provided in some manner, down to the bit level where necessary and are catered for by MADEL.

For example, real numbers are described using the MADEL `REAL` type whose defaults describe ANSI/IEEE 754 floating point numbers or using a generalised form of the `REAL` type: this type, to be used for any non ANSI/IEEE 754 real numbers, provides the capability to fully describe a real number physical layout in terms of mantissa, exponent, base, etc; thereby allowing to compute at the destination the real number value from:

$$ValueOfRealNumber = -1^{Sign} \times Mantissa \times Base^{Exponent-Bias}$$

**Real Time Data Selection**

The purpose of this modification to ASN.1 is to allow the description of situations where one possibility has to be selected among several alternatives at the time the physical data is being received. Such situations are typical in space related applications: for example the value of the first byte in the physical data could indicate the format of satellite tracking measurement samples -- Doppler/Ranging/Meteo --) or a layout word at the beginning of a housekeeping telemetry frame would indicate whether `commands acknowledgement` or `memory dumps` or `attitude parameters` are contained in that particular frame received on ground. Thus, the format of a piece of data typically is dependent upon a value of another piece of data (called the *discriminant*) and it should be possible to describe this aspect.

It has been felt that this ability is fundamental to existing space related data products and since at present ASN.1 does not have it, MADEL has been designed to support it: to this end, the `SELECT` type has been introduced. Thus with MADEL, the discriminant conveyed within the physical data can be processed in order to dictate the branch to be selected, without needing any encoding/decoding mechanism. Furthermore, the type of the discriminant itself must also be specified, for example, `INTEGER`, `REAL`, `IA5String`. The format of the `SELECT` statement in a MADEL description would be:

```
Packet ::= SELECT PacketType {          -- PacketType is the discriminant
                              "AX" : AuxiliaryPacket,
                              "HK" : HouseKeepingPacket,
```

```
                    "NS"  : NormalSciencePacket,
                    "BS"  : BurstSciencePacket
                }
```

together with the following discriminant definition:

```
PacketType ::= IA5String(SIZE (2))
```

**Implementation of the MADEL Interpreter**

The task of interpreting a MADEL description together with an instance of the corresponding physical data is achieved (and demonstrated) by prototype software called the MADEL Interpreter. An overview of the MADEL Interpreter architecture is shown in Figure 2: this shows the main processes that are involved and the critical data structures.

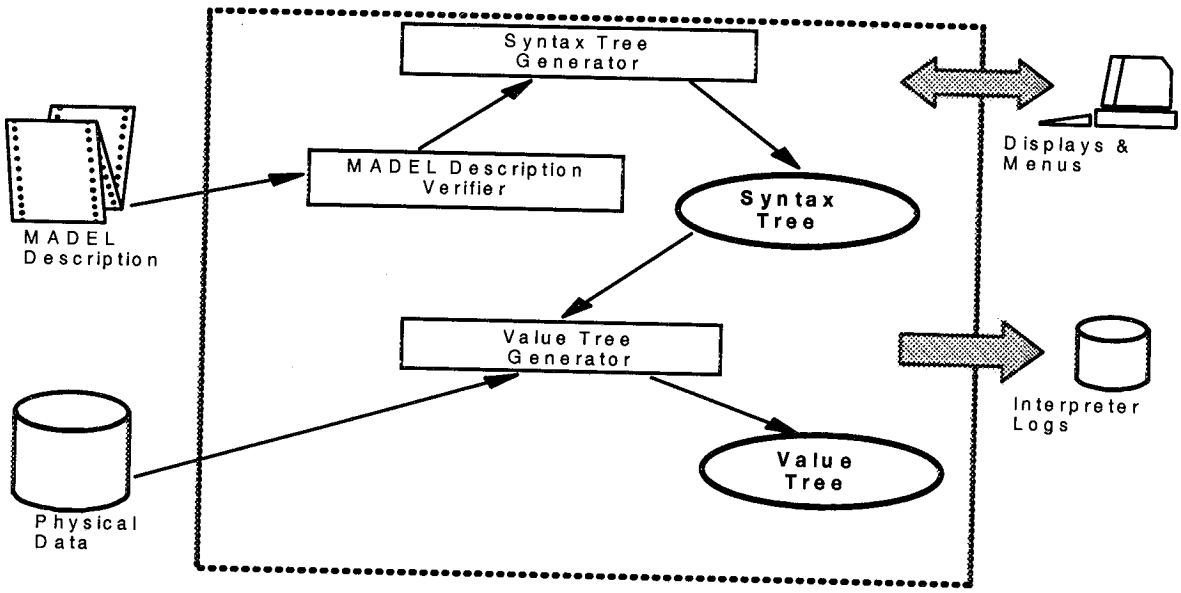


Figure 2: **MADEL Interpreter Software Architecture**

**Running the MADEL Interpreter**

• The user feeds into the MADEL Interpreter a MADEL description of his data and the physical data file. The MADEL description is verified for correctness by the MADEL Description Verifier module.

• The verified MADEL description is then parsed by the Syntax Tree Generator and a syntax tree of the data structure defined in the MADEL description is built internally. This internal tree represents the full syntax description that is defined including all possible choices or selections. Figure 3 shows a schematic of such a tree for a simple data structure: Element **A** is defined as a sequence of element **B** (an integer), followed by **C** (an octet) followed by **D** (a selection) which is dependent upon the value at reception time of the integer **B** (the discriminant). If **B**=1 then **D** will be **P** (an integer), if **B**=2 then **D** will be **Q** (a sequence) and if **B**=3 then **D** will be **R** (a real). Finally if **Q** is selected then this will be a sequence of **X** (an integer) followed by **Y** (a real).
• The Value Tree Generator then walks down the syntax tree reading the corresponding physical data driven by the syntax tree. It sees at the top of the syntax tree that the first element is a sequence, this corresponds to no actual physical data, so it creates the top node of the value tree with no data, just a flag indicating a sequence. It then goes on to read the elements of the sequence; firstly an integer, so bits are read from the physical data

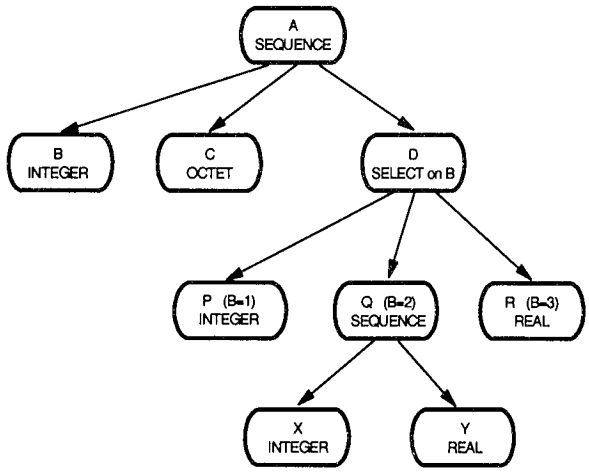


Figure 3: **Schematic of the Syntax Tree Generated**

and the Value Tree Generator adds a node to the value tree which it fills with the **local representation** of the actual value. Following this, the octet corresponding to **C** is read in the same manner. Again the select element **D** corresponds to no physical data, but the selection is dependent upon the value of **B**, so the value of **B** already stored in the value tree is accessed and the relevant branch of the syntax tree taken. Say **B** was 2, then branch **Q** is selected, so this branch of the syntax tree is followed and the corresponding branch of the value tree created. The resultant value tree is shown schematically in Figure 4. Eventually this tree will store the local representation of all selected physical data, and the user's application can proceed with stage 2 of the model (see Figure 1) or access the data using conventional methods. This link to the local representation of the values received in the physical data fulfils the task defined in stage 1 of the model.

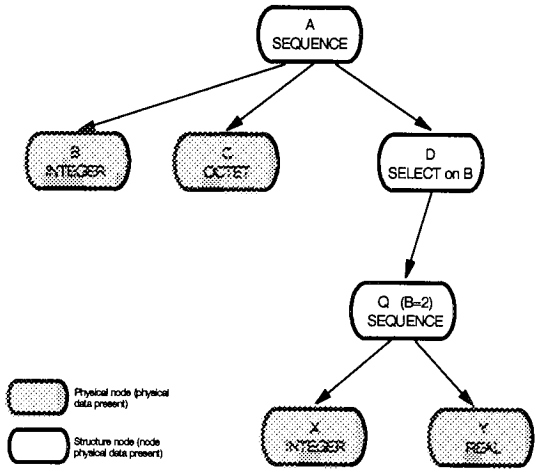


Figure 4: **Schematic of the Value Tree Generated**

### Implementation and Support Tools

The MADEL Interpreter was developed on an IBM PC running MS-DOS, with Borland-C++ and MKS Lex & Yacc as major development tools. It is written in the C language.

Test data sets, along with their corresponding MADEL descriptions must be generated for test purposes. To this end support tools have been developed. A MADEL description is created with a normal text editor and fed into the MADEL Interpreter in ASCII text form. The physical data can be created interactively and a user interface mechanism has been integrated so that the user be provided with facilities to (i) select a MADEL description, then (ii) be guided in the production of associated physical test data.

### Conclusions

• The study has shown the feasibility of the CCSDS concept and the analysis performed while developing MADEL has shown that, if some simple defaults and extensions are adopted (as have been defined), ASN.1 can be used satisfactorily as the basis for a suitable DDL. In particular it can fulfil the fundamental need for real time selection of data in the description of space related data. It should be noted that an important facet of the work was that MADEL, in terms of its syntax, has been kept very close to ASN.1 (as defined in the ISO 8824 standard).

• Prototype software (the MADEL Interpreter) was developed which demonstrated that typical space related data formats can be interchanged between heterogeneous computer environments and interpreted. In turn, this "proof of concept" development and testing helped to improve the overall data interchange model and to identify possible future additions to MADEL. User interface aspects were also studied.

• The MADEL Interpreter could become the basis of a set of tools for supporting data interchanges. Data product definitions or interface definitions written in MADEL could be generated and given to all users, thus avoiding the need for paper Interface Control Documents or paper format definitions. This would ensure that all users have a unique, correctly coded product description.

• In the longer term, stage 3 of the data interchange model should be looked at with the objective of defining standardised approaches and identifying common software services for the processing of advanced semantics.

## EXAMPLES

### Example 1: Interchanging Real Numbers from VAX to IBM

```
-- Hexadecimal dumps of VAX real numbers were generated on a VAX/VMS computer.
-- The corresponding MADEL description was created (this file) and then processed
-- by the MADEL Interpreter on an IBM PC to interpret the corresponding VAX data.

-- GFloatNumber structure on VAX
--    Word_1   {Sign:1 , Exp:11 , Mant:4}
--    Word_2    Word_3   Word_4 :   all {Mant:16}
-- FFloatNumber structure on VAX
--    Word_1   {Sign:1 , Exp:8 , Mant:7}
--    Word_2   {Mant:16}

-- Case 1
-- GFloatDump   Value =  -2.0000       Phys. Hex.    = C020 0000 0000 0000
-- FFloatDump   Value =  -2.0000       Phys. Hex.    = C100 0000

-- Case 2
-- GFloatDump   Value = -64071.5000    Phys. Hex.    = C10F 48F0 0000 0000
-- FFloatDump   Value = -64071.5000    Phys. Hex.    = C87A 4780

-- MADEL Definition starts here:

VaxVmsReals DEFINITIONS  ::=
BEGIN

List-of-VaxNumbers ::= SEQUENCE SIZE( 2 ) OF Numbers      -- 2 cases in this example

Numbers ::=  SEQUENCE {
                       GFloatNumber,
                       FFloatNumber
                      }

GFloatNumber ::= REAL {
                       BIAS (1025),          -- bias value
                       COMPLEMENT(0),        -- 0, 1 or 2's complement indicator
                       BASE(2),              -- base value
                       MANTISSA_MODE( 1),    -- algorithm for mantissa evaluation
                       MANTISSA(12,63),      -- mantissa bit positions
                       EXPONENT( 1,11)       -- exponent bit positions
                      }

FFloatNumber ::= REAL {              -- using ANSI/IEEE 754 defaults for REAL type
                       BIAS( 129)    -- except bias which is VAX specific
                      }

END
```

**MADEL Interpreter Execution Report:**

```
Symbol Name ...:  GFloatNumber
Size ..........:  64     Type Name ...:  REAL
Exponent_bits = (1..11),    Mantissa_bits = (12..63),
```

```
Complement = 0,   Base = 2,   Bias = 1025,   Mantissa_Mode = 1
PHYSICAL REPRESENTATION .: 11000000 00100000 00000000 00000000
                           00000000 00000000 00000000 00000000
 HOST REAL NUMBER = -2
```

---

```
Symbol Name ...:  FFloatNumber
Size ..........:  32    Type Name ...:  REAL
Exponent_bits = (1..8),   Mantissa_bits = (9..31),
Complement = 0,   Base = 2,   Bias = 129,   Mantissa_Mode = 1
PHYSICAL REPRESENTATION .: 11000001 00000000 00000000 00000000
 HOST REAL NUMBER = -2
```

---

```
Symbol Name ...:  GFloatNumber
Size ..........:  64    Type Name ...:  REAL
Exponent_bits = (1..11),   Mantissa_bits = (12..63),
Complement = 0,   Base = 2,   Bias = 1025,   Mantissa_Mode = 1
PHYSICAL REPRESENTATION .: 11000001 00001111 01001000 11110000
                           00000000 00000000 00000000 00000000
 HOST REAL NUMBER = -64071.5
```

---

```
Symbol Name ...:  FFloatNumber
Size ..........:  32    Type Name ...:  REAL
Exponent_bits = (1..8),   Mantissa_bits = (9..31),
Complement = 0,   Base = 2,   Bias = 129,   Mantissa_Mode = 1
PHYSICAL REPRESENTATION .: 11001000 01111010 01000111 10000000
 HOST REAL NUMBER = -64071.5
```

---

## Example 2: Real Time Data Selection

```
-- This example is a special case of the data format discussed in Figure 3
-- and illustrates further the dynamics of data interpretation at the destination

-- MADEL Definition starts here:
Example-of-DataRealTimeSelection  DEFINITIONS  ::=
BEGIN

A ::= SEQUENCE { B, C, D }

B ::= INTEGER                -- specification of the discriminant
C ::= OCTET STRING
D ::= SELECT B {             -- B is discriminant for selecting
                  1 :  P,
                  2 :  Q,
                  3 :  R
                }
P ::= INTEGER
Q ::= SEQUENCE  { X, Y }
R ::= REAL
X ::= INTEGER     -- description of the lowest level elements in
Y ::= REAL        -- the data format A
END
```

**MADEL Interpreter Execution Report:**

---

```
Symbol Name ...:  B
Size ..........:  16    Type Name ...:  INTEGER
PHYSICAL REPRESENTATION .: 00000000 00000010
 INTEGER VALUE =  2
```

---

```
Symbol Name ...:  C
Size ..........:  1    Type Name ...:  OCTET
PHYSICAL REPRESENTATION .: 00101010
 OCTET STRING VALUE = '*'
```

---

```
Symbol Name ...:  D
Size ..........:  16    Type Name ...:  SELECT
```

```
PHYSICAL REPRESENTATION .:     see symbol  B
 SELECTED BRANCH =  2
```

---

```
Symbol Name ...:  X
Size ..........:  16     Type Name ...:  INTEGER
PHYSICAL REPRESENTATION .: 00000000 10000001
 INTEGER VALUE =  129
```

---

```
Symbol Name ...:  Y
Size ..........:  32     Type Name ...:  REAL
Exponent_bits = (1..8),    Mantissa_bits = (9..31),
Complement = 0,    Base = 2,    Bias = 128,    Mantissa_Mode = 1
PHYSICAL REPRESENTATION .: 01000000 01000000 00000000 00000000
 HOST REAL NUMBER = 1.5
```

---

## REFERENCES


[1] "Information technology - Open Systems Interconnection - Specification of Abstract Syntax Notation (ASN.1)", ISO 8824, Second edition, ISO/IEC 8824:1990(E), International Organisation for Standardisation, December 1990.

[2] "Recommendation for Space Data System Standards: Parameter Value Language -- A Tutorial", CCSDS 641.0-G-1, Green Book, Consultative Committee for Space Data Systems, May 1992.